**DataBase Project 2 보고서**

**12201768 이근영**

**What you implemented and what you have not**

B+-tree를 만드는 기능, 텍스트 파일에 담겨진 키값과 벨류값을 B+-tree에 입력하는 기능, 텍스트 파일 속 키값을 읽어 B+-tree속 벨류를 찾아 키값과 함께 출력파일에 넣는 기능, 텍스트 파일 속 키값의 최대 값, 최소 값을 받아 해당 범위 내 키값을 벨류와 함께 출력파일에 담는 기능, 모든 레벨의 키값들을 차례대로 형식에 맞게 출력파일에 넣는 기능을 구현하였다.

**Brief explanation of your implementation**

기본적으로 블록단위로 파일에서 불러와서 작업을 수행하였다. 또한 새로쓰기는 wb, 덮어쓰기는 r+b, 이어쓰기는 a+b, 읽기만 할 때는 rb로 fopen시에 기능에 따라 다른 모드를 사용하였다. 또한 헷갈리지 않게 하나의 엔티티를 두개의 엔티티로, 즉 포인터와 키값을 그냥 하나의 엔티티로 생각하였다. 즉 BlockSize/4개의 엔티티가 있다고 생각하였고 아래 설명의 엔티티도 이를 뜻한다.

Create: b+tree 클래스 객체에 헤더, 블록크기, 블록 속 엔티티의 수 등을 변수로 가지고 생성자를 호출하면 초기화한다. 생성자 중에서 파일이름, 블록크기를 파라미터로 받는 생성자는 헤더의 첫번째 원소를 블록크기로 바꾸고 나머지를 0으로 초기화 한 뒤 파일이름에 해당하는 파일을 fopen의 wb모드로 열어 파일을 비우고 헤더 정보 12바이트를 바이너리형식으로 넣으며 b+tree를 만든다. 해당 이름의 파일이 없으면 생성한다. 파일 이름만을 파라미터로 받는 생성자는 파일이름에 해당하는 파일의 헤더부분을 읽어와 클래스 속 헤더를 초기화 하는 함수로 create 외에 다른 함수를 수행하기 전에 클래스를 초기화 하기 위해 수행한다.

Insertion: 파일을 fgetc로 한자씩 받아와 “,”와 “\n”을 제외한 문자를 문자열 변수에 담고 int로 변환하여 함수에 집어넣었다. 일단 만약 트리의 루트가 0 즉, 아무 블록도 없을 때에는 입력받은 키값과 벨류값을 0번째 1번째 인자로 갖는 블록을 생성한 뒤 파일을 fopen의 a+b모드로 열어 헤더 뒤에 블록을 입력해주었다.

루트에서 시작하여 키값에 알맞은 leaf블록에 도달할 때까지 블록을 타고 내려가며 지나간 블록, 즉 해당 키값이 들어가기 위해 거쳐야 하는 블록들의 번호를 배열에 저장하는 함수인 searchB함수를 만들어 insertion에 활용하였다. 또한 애초에 클래스 내 블록의 크기를 입력 받은 크기보다 2칸 더 크게 만들었다. 왜냐하면 해당 블록이 꽉찬 상태인 경우 나누어줄 때 편하게 하기 위해서 이다. 만약 블록 속 엔티티가 7개가 최대라면 9개를 선언한 뒤 추가되는 키값을 알맞은 자리에 넣고 만약 9칸이 다 꽉찼다면 반으로 나누어주는 방법을 사용하였다. 이 때 leaf블록은 마지막 포인터를 항상 마지막칸에 오도록 뒤로 밀어 사용하였다.

블록들이 분할되며 부모노드로 올라갈 때에는 searchB를 통해 받아두었던 블록번호 배열을 사용하여 위로 올라가며 분할, 추가를 진행하였다. 만약 배열의 마지막 원소, 즉 루트 블록도 분할하여야 하면 분할하고 새로운 루트블록을 만든 뒤 헤더를 고치고 파일 속 헤더를 수정하고 추가된 블록을 넣었다.

Search: insertion시에 만들어 두었던 searchB를 약간 수정하여 키값이 들어갈 만한 leaf블록을 찾고 해당 블록을 읽어와 키값와 벨류값을 찾아 출력파일에 작성하였다.

Search Range: search를 활용하여 위치를 찾고 해당 키값의 최대크기 보다 큰 값이 나올 때 까지 이동하며 문자열에 더하고 출력파일에 문자열을 입력하였다. 해당 블록을 다 찾았는데 블록 속 마지막 키값이 설정한 한계보다 작다면 leaf블록의 마지막 엔티티에 있는 옆 블록번호를 이용해 옆 블록을 불러와 찾는 것을 반복하도록 하였다.

PrintAll: 문제 설명에서는 level 1까지만 출력하라고 하였지만 문제 설명을 제대로 읽지 않고 짜느라 모든 level이 출력되도록 만들어버렸다. 일단 2차원 벡터를 선언하고 첫 칸에 루트블록을 저장한 뒤 루트블록부터 하나씩 불러오며 해당 블록이 가리키는 다른 블록번호를 벡터의 다음 칸에 (다음 벡터에)넣고 또 들어간 블록을 하나씩 다 불러오며 가리키는 블록들을 벡터에 넣도록 하여 이차원 벡터가 각 level의 블록들의 번호를 저장하도록 하였다.

Ex) v[0]:루트블록번호, v[1]:level1의 블록번호들, v[2]:level2의 블록번호들…

그 뒤 level0부터 블록번호에 맞는 블록을 받아와 출력파일에 필요한 내용을 입력하였다.

**How to compile and run**

Visual studio로 cpp파일을 열고 실행시키면 debug 폴더 내에 exe파일이 생성된다.

exe실행 파일과 사용할 input,output파일들을 한 폴더에 넣고 cmd로 해당 디렉토리 위치로 간 뒤

btree.exe c btree.bin [Size] 를 입력하면 해당 블록사이즈를 가지는 b+tree파일이 만들어진다.

btree.exe i btree.bin [input.txt] 를 입력하면 input.텍스트 파일 속 인자들이 b+tree에 넣어진다.

btree.exe s btree.bin [input.txt] [output.txt] 를 입력하면 input텍스트 파일 속 키값에 알맞은 값들이 output텍스트 파일에 들어간다.

btree.exe r btree.bin [input.txt] [output.txt] 를 입력하면 input텍스트 파일 속 키값에 알맞은 범위의 값들이 값들이 output텍스트 파일에 들어간다.

btree.exe p btree.bin [output.txt] 를 입력하면 트리의 전체 모습이 나온다.

**Talk about your experience of doing this project**

파일입출력을 잘 다루지 못해 익숙해지는데 어려움이 있었다. Insertion을 구현할 때 이상한 값이 들어가거나 잘못 읽거나 하는 등 문제가 가장 많았고 오랜 시간이 걸렸다. 하지만 몇 시간 동안 코드랑 씨름하면서 결국에는 완성 하였을 때, 쾌감을 느꼈다. searchRange를 구현했는데 출력도 정상적으로 잘 되는데 이상한 오류 안내창이 뜨는 버그가 있다. 원인을 찾지 못했고 오류창만 뜰 뿐 정상적으로 작동하길래 그냥 두었다. 이론으로만 배웠던 B+-tree를 직접 만들어보니 더 자세히 알게 되는 느낌이었다.

**Write your available contact information such as phone**

Email: [zuzu5313@naver.com](mailto:zuzu5313@naver.com)

Phone: 010-5720-5314